Software Paradigms SS 2015 2

Nachname	Vorname	Matrikelnummer
LORENZ	Peter	
ZIKO	Haris	

Example 1

δ `count = if eq?(x, nil) then 0 else plus(count(rest(x)),1)`

We define $f_d(\omega(\underline{(}x)))$ as "Golden Device", a function (L $\to \mathbb{N}$).

$$|\omega(\underline{(}x))\;| = \begin{cases} 0 & eq?(\omega(\underline{x}), []) \\ plus(count(rest(x)), 1) & other \end{cases}$$

Idea - if δcount is correct $|\omega(\underline{x})\;| = n$ ($n \in \mathbb{N}$), then is δcount is also correct $|\omega(\underline{x})| = n + 1$.

Lemma: $\forall \omega(x) \in L$, $|\omega(x)\;| \leq n$: $I(\delta, \omega, count(x) = f_d(\omega(x))$

Base: $\omega(\underline{x}) \in L, |\omega(\underline{x})\;| = 0$
$I(\delta, \omega, \underline{count(x)})$
$= I(\delta, \omega, \underline{if\ eq?(x, nil)\ then\ 0\ else\ plus(count(rest(x)),1)})$

NR:
$I(\delta, \omega, \underline{eq?(x, nil)})$
$= eq?(I(\delta, \omega, \underline{x}), I(\delta, \omega, \underline{nil}))$
$= eq?(\omega(\underline{(}x)), [])$
$= eq?([], [])$
$= T$

$I(\delta, \omega, \underline{0})$
Since $|\omega(x)\;| = 0$ must be $\omega(\underline{x}) = []$.
$= I(\delta, \omega, \underline{nil}) = [] = f_d([]) = f_d(\omega(\underline{x}))$

Step: $\omega(x) \in L$, $|\omega(\underline{(}x))\;| = n + 1$, $n \in \mathbb{N}$
$I(\delta, \omega, \underline{count(x)})$
$= I(\delta, \omega, \underline{if\ eq?(x, nil)\ then\ 0\ else\ plus(count(rest(x)),1)})$

NR:
$I(\delta, \omega, \underline{eq?(x, nil)})$
$= eq?(I(\delta, \omega, \underline{x}), I(\delta, \omega, \underline{nil}))$
$= eq?(\omega(\underline{x}), [])$
$= eq?(x, []) = F$

$= I(\delta, \omega, \underline{plus(count(rest(x)), 1)})$
$\omega'(\underline{x}) = I(\delta, \omega, \underline{rest(x)}) = rest(I(\delta, \omega, \underline{x})) = rest(\omega(\underline{x}))$
$= I(\delta, \omega', \underline{count(x)})$

The induction hypothesis is valid for this environment. We apply the the induction hypothesis:

$= f_d(\omega'(\underline{x}))$
$= f_d(\text{rest}(\omega(\underline{x})))$

Example 2

```
δ gcd = if gt?(a,b) then gcd2(a,b) else gcd2(b,a)
δ gcd2 = if eq?(b,0) then a else gcd2(b, mod(a,b))
δ mod = if gt?(a,b) then mod(minus(a,b), b) else if eq?(a,b) then 0 else a
```

Example 3

```
δ bijunct = if atom?( first(x) ) then
    build( bijunct2(first(x), first(y)), bijunct(rest(x),rest(y)) )
    else nil
δbijunct2 = if is0?(x) then
    if is0?(y) then 1 else if is1?(y) then 0 else
    if is1?(x) then
    if is0?(y) then 0 else if is1?(y) then 1
```

Golden Device: $f_d(\omega(l_x), \omega(l_y))$, B × B → B

$$\sim(I_V(\omega,l_x) \oplus I_V(\omega,l_y)) = \begin{cases} l_x & l_y \\ 0 & 0 = 1 \\ 0 & 1 = 0 \\ 1 & 0 = 0 \\ 1 & 1 = 1 \end{cases}$$

Lemma: $\forall\omega \in$ ENV : I_ε (δ, ω, bijunct(x,y)) = $\sim(I_V(\omega,\text{x}) \oplus I_V(\omega,\text{y}))$ if x, y $\in$ L, $\omega \in$ENV

Proof: Let ω(x) = $[l_1, \ldots, l_k]$ with $\forall l_i \mathbb{B}$ and ω(y) = $[l_1, \ldots, l_k]$ with $\forall l_i$ $\mathbb{B}$

$I_\varepsilon(\delta, \omega, \text{bijunct(x, y)})$
$= I_\varepsilon(\delta, \omega', \text{bijunct(x1, y1)})$ with &
$\omega'(\text{x1}) = (\delta, \omega, \text{x}) = \omega(\text{x}) = [l_1, \ldots, l_n]$
$\omega'(\text{y1}) = (\delta, \omega, \text{y}) = \omega(\text{y}) = [l_1, \ldots, l_n]$
$= \sim(\omega'(\text{x1}) \oplus \omega\text{'(y1)})$ [Lemma]

Base: $\omega(\underline{x}) = []$, $\omega(\underline{y}) = []$, $|\omega(\text{x})| = |\omega(\text{y})| = 1$

= I(δ, ω, $\underline{\text{bijunct(x,y)}}$)
= I(δ, ω', $\underline{\text{if atom?(first(x1) then build(bijunct2(first(x), first(y)), bijunct(rest(x),rest(y)))))}}$
$\omega'(\underline{\text{x1}})$ = I(δ, ω, $\underline{x}$) = []
NR: I(δ, ω', atom?($\underline{\text{first(x1)}}$) = atom?(I($\delta$, ω, $\underline{x}$)) = atom? (first([]) = atom?([]) = F
= I(δ, ω, $\underline{\text{nil}}$) = []

Step: ω(x) = [] + $[l_1, \ldots, l_n]$, $|\omega(\text{x})\ | = |\ \omega(y)\ |$ = n + 1

I(δ, ω, $\underline{\text{bijunct(x, y))}}$)
= I(δ, ω, $\underline{\text{if atom?(first(x)) then build(bijunct2(first(x), first(y)), bijunct(rest(x),rest(y))))}}$)

case distinction towards atom?(first(x)):

1. case: atom?(first(ω(x))) = F
   I(δ, ω, atom?(first(x))) = atom?(I(δ, ω, $\underline{\text{first(x)}}$)) = atom?(first(I($\delta$,$\omega$, $\underline{\text{x}}$))) = F
   = I(δ, ω, $\underline{\text{nil}}$) = []
2. case: atom?(first(ω(x)) = T
   NR: I(δ, ω, atom?(first(ω(x)))) = atom?(I(δ, ω, first(x))) = atom?(first(I(δ, ω, $\underline{\text{x}}$))) = T
   =I(δ, ω, $\underline{\text{if atom?(first(x)) then}}$ $\underline{\text{build(bijunct2(first(x), first(y)), bijunct(rest(x),rest(y))) else nil}}$)

   Since | ω($\underline{\text{x}}$) | ¿ 1, consequently ω($\underline{\text{x}}$) $\neq$ []
   = I(δ, ω, $\underline{\text{build(bijunct2(first(x), first(y)), bijunct(rest(x),rest(y)))}}$)
   NR: I(δ, ω, bijunct2(first(x), first(y)) = f_d(ω(x), ω(y))

q.e.d

Example 4

```
def reverse(as : List[Int]) : List[Int] = as match {
  case Nil   => Nil
  case x::xs => reverse(xs):+x
}

def append(as : List[Int], bs : List[Int]) : List[Int] = as match {
  case Nil   => bs
  case x::xs => x::append(xs, bs)
}
```

Hypothesis: reverse(append(as, bs)) = append(reverse(bs), reverse(as))

Base: as with Nil substituted
reverse(append(Nil,bs))
= append(reverse(bs),reverse(Nil)) [inductive hypothesis l.r.]

Step: as with a::as extended
reverse(append(a::as,bs))
= reverse(a::append(as, bs)) [def. append l.r.]
= reverse(append(as, bs)):+a [def. reverse l.r.]
= append(reverse(bs), reverse(as)):+a [induction hypothesis]
= append(reverse(bs), reverse(as):+a) [associative of append]
= append(reverse(bs), reverse(a::as)) [def. reverse r.l.]
q.e.d

Example 5

```
def without(x : Int, to_check : List[Int]) : Boolean = to_check match {
   case head::tail => if(head == x) false else without(x, tail)
   case Nil => true
}

def append(as : List[Int], bs : List[Int]) : List[Int] = as match {
  case Nil   => bs
  case x::xs => x::append(xs, bs)
}
```

Hypothesis: without(x, append(as, bs)) = without(x, as) && without(x, bs)

Base: as with Nil substituted
without(x, append(Nil, bs))
= without(x, Nil) && without(x, bs) [induction hypothesis]

Step: as with a::as extended
without(x, append(a::as, bs))
= without(x, a::append(as, bs)) [def. append l.r.]
= if (a == x) false else without(x, append(as, bs)) [def. without l.r.]
= if (a == x) false else without(x, as) && without(x, bs) [induction hypothesis]
= without(x, a::as) && without(x, bs) [def. without r.l.]
q.e.d

Example 6

```
def sum(as: List[Int]) : Int = as match {
  case Nil => 0
  case x::xs => x + sum(xs)
}

def sum1(as: List[Int], i : Int): Int = as match {
  case Nil => i
  case x::xs => sum1(xs, i + x)
}
```

Hypothesis: sum(as) = sum1(as, 0)

Base: as as Nil substituted
sum1(Nil, 0)
= 0 [def. sum1 l.r.]
= sum(Nil) [def. sum r.l.]

Step: as with a::as extended
sum1(a::as, 0)
= sum1(as, 0 + a) [def. sum1 l.r.]
= sum1(as, a) [arithmetic]
= a + sum(as) [induction hypothesis r.l.]
= sum(a::as) [def. sum r.l.]
q.e.d